# ZIP (file format)

**ZIP** is an archive file format that supports lossless data compression. A ZIP file may contain one or more files or directories that may have been compressed. The ZIP file format permits a number of compression algorithms, though DEFLATE is the most common. This format was originally created in 1989 and was first implemented in PKWARE, Inc.'s PKZIP utility,[2] as a replacement for the previous ARC compression format by Thom Henderson. The ZIP format was then quickly supported by many software utilities other than PKZIP. Microsoft has included built-in ZIP support (under the name "compressed folders") in versions of Microsoft Windows since 1998 via the "Windows Plus!" addon for Windows 98. Native support was added as of the year 2000 in Windows ME. Apple has included built-in ZIP support in Mac OS X 10.3 (via BOMArchiveHelper, now Archive Utility) and later. Most free operating systems have built in support for ZIP in similar manners to Windows and Mac OS X.

ZIP files generally use the file extensions `.zip` or `.ZIP` and the MIME media type `application/zip`.[1] ZIP is used as a base file format by many programs, usually under a different name. When navigating a file system via a user interface, graphical icons representing ZIP files often appear as a document or other object prominently featuring a zipper.

## History[edit]

The `.ZIP` file format was designed by Phil Katz of PKWARE and Gary Conway of Infinity Design Concepts. The format was created after Systems Enhancement Associates (SEA) filed a lawsuit against PKWARE claiming that the latter's archiving products, named PKARC, were derivatives of SEA's ARC archiving system.[3] The name "zip" (meaning "move at high speed") was suggested by Katz's friend, Robert Mahoney.[4] They wanted to imply that their product would be faster than ARC and other compression formats of the time.[4] The earliest known version of *.ZIP File Format Specification* was first published as part of PKZIP 0.9 package under the file APPNOTE.TXT in 1989.[*citation needed*] By distributing the zip file format within APPNOTE.TXT, compatibility with the zip file format proliferated widely on the public Internet during the 1990s.[5]

PKWARE and Infinity Design Concepts made a joint press release on February 14, 1989, releasing the `.ZIP` file format into the public domain.[6][7][8][9][10]

### Version history[edit]

The .ZIP File Format Specification has its own version number, which does not necessarily correspond to the version numbers for the PKZIP tool, especially with PKZIP 6 or later. At various times, PKWARE has added preliminary features that allow PKZIP products to extract archives using advanced features, but PKZIP products that create such archives are not made available until the next major

release. Other companies or organizations support the PKWARE specifications at their own pace.

The .ZIP file format specification is formally named "APPNOTE - .ZIP File Format Specification" and it is published on the PKWARE.com website since the late 1990s.[11] Several versions of the specification were not published. Specifications of some features such as BZIP2 compression, strong encryption specification and others were published by PKWARE a few years after their creation. The URL of the online specification was changed several times on the PKWARE website. A summary of key advances in various versions of the PKWARE specification:

- 2.0: (1993)[1] File entries can be compressed with DEFLATE and use traditional PKWARE encryption (ZipCrypto).
- 2.1: (1996) Deflate64 compression
- 4.5: (2001)[12] Documented 64-bit zip format.
- 4.6: (2001) BZIP2 compression (not published online until the publication of APPNOTE 5.2)
- 5.0: (2002) SES: DES, Triple DES, RC2, RC4 supported for encryption (not published online until the publication of APPNOTE 5.2)
- 5.2: (2003)[13][14] AES encryption support for SES (defined in APPNOTE 5.1 that was not published online) and AES from WinZip ("AE-x"); corrected version of RC2-64 supported for SES encryption.
- 6.1: (2004)[15] Documented certificate storage.
- 6.2.0: (2004)[16] Documented Central Directory Encryption.
- 6.3.0: (2006)[17] Documented Unicode (UTF-8) filename storage. Expanded list of supported compression algorithms (LZMA, PPMd+), encryption algorithms (Blowfish, Twofish), and hashes.
- 6.3.1: (2007)[18] Corrected standard hash values for SHA-256/384/512.
- 6.3.2: (2007)[19] Documented compression method 97 (WavPack).
- 6.3.3: (2012)[20] Document formatting changes to facilitate referencing the PKWARE Application Note from other standards using methods such as the JTC 1 Referencing Explanatory Report (RER) as directed by JTC 1/SC 34 N 1621.
- 6.3.4: (2014)[21] Updates the PKWARE, Inc. office address.
- 6.3.5: (2018)[22] Documented compression methods 16, 96 and 99, DOS timestamp epoch and precision, added extra fields for keys and decryption, as well as typos and clarifications.
- 6.3.6: (2019)[23] Corrected typographical error.
- 6.3.7: (2020)[24] Added Zstandard compression method ID 20. • 6.3.8: (2020)[25] Moved Zstandard compression method ID from 20 to 93, deprecating the former. Documented method IDs 94 and 95 (MP3 and XZ respectively).
- 6.3.9: (2020)[26] Corrected a typo in Data Stream Alignment description.

WinZip, starting with version 12.1, uses the extension `.zipx` for ZIP files that use compression methods newer than DEFLATE; specifically, methods BZip, LZMA, PPMd, Jpeg and Wavpack. The last 2 are applied to appropriate file types when "Best method" compression is selected.[27][28]

### Standardization[edit]

In April 2010, ISO/IEC JTC 1 initiated a ballot to determine whether a project should be initiated to create an ISO/IEC International Standard format compatible with ZIP.[29] The proposed project, entitled *Document Packaging*, envisaged a ZIPcompatible 'minimal compressed archive format' suitable for use with a number of existing standards including OpenDocument, Office Open XML and EPUB.
In 2015, ISO/IEC 21320-1 "Document Container File — Part 1: Core" was published which states that "Document container files are conforming Zip files". It requires the following main restrictions of the ZIP file format:[30]

- Files in ZIP archives may only be stored uncompressed, or using the "deflate" compression (i.e. compression method may contain the value "0" - stored or "8" - deflated).
- The encryption features are prohibited.
- The digital signature features (from SES) are prohibited.
- The "patched data" features (from PKPatchMaker) are prohibited.
- Archives may not span multiple volumes or be segmented.

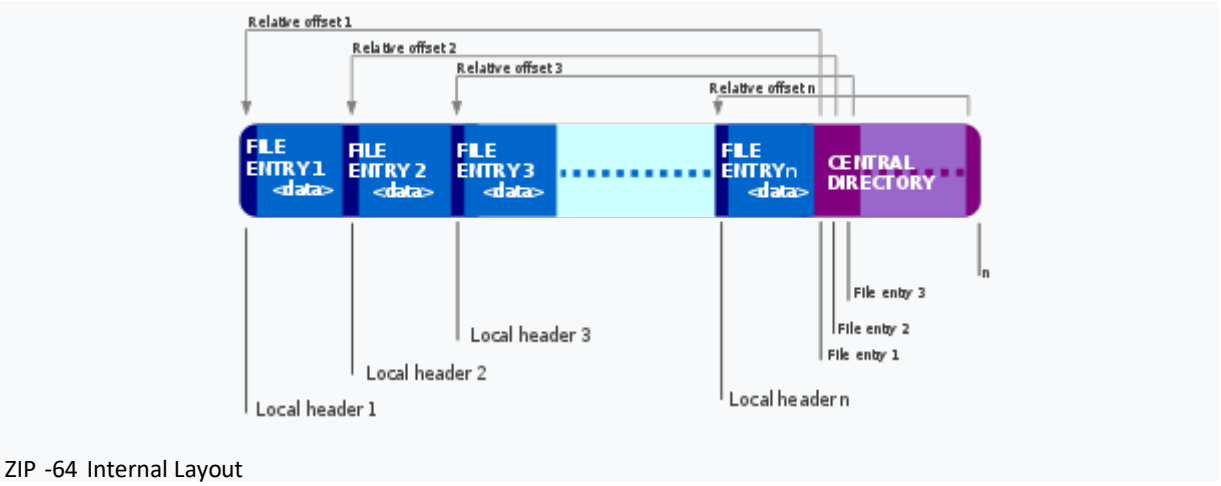
## Design[edit]

`.ZIP` files are archives that store multiple files. ZIP allows contained files to be compressed using many different methods, as well as simply storing a file without compressing it. Each file is stored separately, allowing different files in the same archive to be compressed using different methods. Because the files in a ZIP archive are compressed individually, it is possible to extract them, or add new ones, without applying compression or decompression to the entire archive. This contrasts with the format of compressed tar files, for which such random-access processing is not easily possible.

A directory is placed at the end of a ZIP file. This identifies what files are in the ZIP and identifies where in the ZIP that file is located. This allows ZIP readers to load the list of files without reading the entire ZIP archive. ZIP archives can also include extra data that is not related to the ZIP archive. This allows for a ZIP archive to be made into a self-extracting archive (application that decompresses its contained data), by prepending the program code to a ZIP archive and marking the file as executable. Storing the catalog at the end also makes possible hiding a zipped file by appending it to an innocuous file, such as a GIF image file.

The `.ZIP` format uses a 32-bit CRC algorithm and includes two copies of the directory structure of the archive to provide greater protection against data loss.

### Structure[edit]




ZIP -64 Internal Layout

A ZIP file is correctly identified by the presence of an *end of central directory record* which is located at the end of the archive structure in order to allow the easy appending of new files. If the end of central directory record indicates a non-empty archive, the name of each file or directory within the archive should be specified in a *central directory* entry, along with other metadata about the entry, and an offset into the ZIP file, pointing to the actual entry data. This allows a file listing of the archive to be performed relatively quickly, as the entire archive does not have to be read to see the list of files. The entries within the ZIP file also include this information, for redundancy, in a *local file header*. Because ZIP files may be appended to, only files specified in the central directory at the end of the file are valid. Scanning a ZIP file for local file headers is invalid (except in the case of corrupted archives), as the central directory may declare that some files have been deleted and other files have been updated.

For example, we may start with a ZIP file that contains files A, B and C. File B is then deleted and C updated. This may be achieved by just appending a new file C to the end of the original ZIP file and adding a new central directory that only lists file A and the new file C. When ZIP was first designed, transferring files by floppy disk was common, yet writing to disks was very time-consuming. If you had a large zip file, possibly spanning multiple disks, and only needed to update a few files, rather than reading and re-writing all the files, it would be substantially faster to just read the old central directory, append the new files then append an updated central directory.

The order of the file entries in the central directory need not coincide with the order of file entries in the archive.

Each entry stored in a ZIP archive is introduced by a *local file header* with information about the file such as the comment, file size and file name, followed by optional "extra" data fields, and then the possibly compressed, possibly encrypted file data. The "Extra" data fields are the key to the extensibility of the ZIP format. "Extra" fields are exploited to support the ZIP64 format, WinZip-compatible AES encryption, file attributes, and higher-resolution NTFS or Unix file timestamps. Other extensions are possible via the "Extra" field. ZIP tools are required by the specification to ignore Extra fields they do not recognize.

The ZIP format uses specific 4-byte "signatures" to denote the various structures in the file. Each file entry is marked by a specific signature. The end of central directory

record is indicated with its specific signature, and each entry in the central directory starts with the 4-byte *central file header signature*.

There is no BOF or EOF marker in the ZIP specification. Conventionally the first thing in a ZIP file is a ZIP entry, which can be identified easily by its *local file header signature*. However, this is not necessarily the case, as this not required by the ZIP specification - most notably, a self-extracting archive will begin with an executable file header.

Tools that correctly read ZIP archives must scan for the end of central directory record signature, and then, as appropriate, the other, indicated, central directory records. They must not scan for entries from the top of the ZIP file, because (as previously mentioned in this section) only the central directory specifies where a file chunk starts and that it has not been deleted. Scanning could lead to false positives, as the format does not forbid other data to be between chunks, nor file data streams from containing such signatures. However, tools that attempt to recover data from damaged ZIP archives will most likely scan the archive for local file header signatures; this is made more difficult by the fact that the compressed size of a file chunk may be stored after the file chunk, making sequential processing difficult. Most of the signatures end with the short integer 0x4b50, which is stored in littleendian ordering. Viewed as an ASCII string this reads "PK", the initials of the inventor Phil Katz. Thus, when a ZIP file is viewed in a text editor the first two bytes of the file are usually "PK". (DOS, OS/2 and Windows self-extracting ZIPs have an EXE before the ZIP so start with "MZ"; self-extracting ZIPs for other operating systems may similarly be preceded by executable code for extracting the archive's content on that platform.)

The `.ZIP` specification also supports spreading archives across multiple file-system files. Originally intended for storage of large ZIP files across multiple floppy disks, this feature is now used for sending ZIP archives in parts over email, or over other transports or removable media.

The FAT filesystem of DOS has a timestamp resolution of only two seconds; ZIP file records mimic this. As a result, the built-in timestamp resolution of files in a ZIP archive is only two seconds, though extra fields can be used to store more precise timestamps. The ZIP format has no notion of time zone, so timestamps are only meaningful if it is known what time zone they were created in.

In September 2007, PKWARE released a revision of the ZIP specification providing for the storage of file names using UTF-8, finally adding Unicode compatibility to ZIP.[31]

## File headers[edit]

All multi-byte values in the header are stored in little-endian byte order. All length fields count the length in bytes.

### Local file header[edit]

| Local file header |
| --- |

| Offset | Bytes | Description[31] |
|---|---|---|
| 0 | 4 | Local file header signature = 0x04034b50 (PK♥♦ or "PK\3\4") |
| 4 | 2 | Version needed to extract (minimum) |
| 6 | 2 | General purpose bit flag |
| 8 | 2 | Compression method; e.g. none = 0, DEFLATE = 8 (or "\0x08\0x00") |
| 10 | 2 | File last modification time |
| 12 | 2 | File last modification date |
| 14 | 4 | CRC-32 of uncompressed data |
| 18 | 4 | Compressed size (or 0xffffffff for ZIP64) |
| 22 | 4 | Uncompressed size (or 0xffffffff for ZIP64) |
| 26 | 2 | File name length (*n*) |
| 28 | 2 | Extra field length (*m*) |
| 30 | *n* | File name |
| 30+*n* | *m* | Extra field |

The extra field contains a variety of optional data such as OS-specific attributes. It is divided into records, each with at minimum a 16-bit signature and a 16-bit length. A ZIP64 local file extra field record, for example, has the signature 0x0001 and a length of 16 bytes (or more) so that two 64-bit values (the compressed and uncompressed sizes) may follow. Another common local file extension is 0x5455 (or "UT") which contains 32-bit UTC UNIX timestamps.

This is immediately followed by the compressed data.

**Data descriptor**[edit]

If the bit at offset 3 (0x08) of the general-purpose flags field is set, then the CRC-32 and file sizes are not known when the header is written. If the archive is in Zip64 format, the compressed and uncompressed size fields are 8 bytes long instead of 4 bytes long (see section 4.3.9.2[32]). The equivalent fields in the local header (or in the Zip64 extended information extra field in the case of archives in Zip64 format) are filled with zero, and the CRC-32 and size are appended in a 12-byte structure (optionally preceded by a 4-byte signature) immediately after the compressed data:

| Data descriptor | | |
|---|---|---|
| **Offset** | **Bytes** | **Description**[31] |
| 0 | 0 or 4 | *Optional* data descriptor signature = 0x08074b50 |
| 0 or 4 | 4 | CRC-32 of uncompressed data |
| 4 or 8 | 4 or 8 | Compressed size |
| 8 or 12 or 20 | 4 or 8 | Uncompressed size |

**Central directory file header**[edit]

The central directory entry is an expanded form of the local header:

| Central directory file header | | |
|---|---|---|
| **Offset** | **Bytes** | **Description**[31] |
| 0 | 4 | Central directory file header signature = 0x02014b50 |
| 4 | 2 | Version made by |
| 6 | 2 | Version needed to extract (minimum) |

| | | |
|---|---|---|
| 8 | 2 | General purpose bit flag |
| 10 | 2 | Compression method |
| 12 | 2 | File last modification time |
| 14 | 2 | File last modification date |
| 16 | 4 | CRC-32 of uncompressed data |
| 20 | 4 | Compressed size (or 0xffffffff for ZIP64) |
| 24 | 4 | Uncompressed size (or 0xffffffff for ZIP64) |
| 28 | 2 | File name length (*n*) |
| 30 | 2 | Extra field length (*m*) |
| 32 | 2 | File comment length (*k*) |
| 34 | 2 | Disk number where file starts |
| 36 | 2 | Internal file attributes |
| 38 | 4 | External file attributes |
| 42 | 4 | Relative offset of local file header. This is the number of bytes between the start of the first disk on which the file occurs, and the start of the local file header. This allows software reading the central directory to locate the position of the file inside the ZIP file. |

| | | |
|---|---|---|
| 46 | *n* | File name |
| 46+*n* | *m* | Extra field |
| 46+*n*+*m* | *k* | File comment |

**End of central directory record (EOCD)**[edit]

After all the central directory entries comes the end of central directory (EOCD) record, which marks the end of the ZIP file:

| End of central directory record (EOCD) | | |
|---|---|---|
| **Offset** | **Bytes** | **Description**[31] |
| 0 | 4 | End of central directory signature = 0x06054b50 |
| 4 | 2 | Number of this disk (or 0xffff for ZIP64) |
| 6 | 2 | Disk where central directory starts (or 0xffff for ZIP64) |
| 8 | 2 | Number of central directory records on this disk (or 0xffff for ZIP64) |
| 10 | 2 | Total number of central directory records (or 0xffff for ZIP64) |
| 12 | 4 | Size of central directory (bytes) (or 0xffffffff for ZIP64) |
| 16 | 4 | Offset of start of central directory, relative to start of archive (or 0xffffffff for ZIP64) |
| 20 | 2 | Comment length (*n*) |
| 22 | *n* | Comment |

This ordering allows a ZIP file to be created in one pass, but the central directory is also placed at the end of the file in order to facilitate easy removal of files from multiple-part *(e.g. "multiple floppy-disk")* archives, as previously discussed.

## Compression methods[edit]

The .ZIP File Format Specification documents the following compression methods: Store (no compression), Shrink (LZW), Reduce (levels 1–4; LZ77 + probabilistic), Implode, Deflate, Deflate64, bzip2, LZMA, WavPack, PPMd, and a LZ77 variant provided by IBM z/OS CMPSC instruction.[33][22] The most commonly used compression method is DEFLATE, which is described in IETF RFC 1951.

Other methods mentioned, but not documented in detail in the specification include: PKWARE DCL Implode (old IBM TERSE), new IBM TERSE, IBM LZ77 z Architecture (PFS), and a JPEG variant. A "Tokenize" method was reserved for a third party, but support was never added.[22]

The word *Implode* is overused by PKWARE: the DCL/TERSE Implode is distinct from the old PKZIP Implode, a predecessor to Deflate. The DCL Implode is undocumented partially due to its proprietrary nature held by IBM, but Mark Adler has nevertheless provided a decompressor called "blast" alongside zlib.[34]

## Encryption[edit]

ZIP supports a simple password-based symmetric encryption system generally known as ZipCrypto. It is documented in the ZIP specification, and known to be seriously flawed. In particular, it is vulnerable to known-plaintext attacks, which are in some cases made worse by poor implementations of random-number generators.[5]

New features including new compression and encryption (e.g. AES) methods have been documented in the ZIP File Format Specification since version 5.2. A WinZipdeveloped AES-based open standard ("AE-x" in APPNOTE) is used also by 7Zip and Xceed, but some vendors use other formats.[35] PKWARE SecureZIP (SES, proprietary) also supports RC2, RC4, DES, Triple DES encryption methods, Digital Certificate-based encryption and authentication (X.509), and archive header encryption. It is, however, patented (see § Strong encryption controversy).[36]

File name encryption is introduced in .ZIP File Format Specification 6.2, which encrypts metadata stored in Central Directory portion of an archive, but Local Header sections remain unencrypted. A compliant archiver can falsify the Local Header data when using Central Directory Encryption. As of version 6.2 of the specification, the Compression Method and Compressed Size fields within Local Header are not yet masked. **ZIP64**[edit]

The original `.ZIP` format had a 4 GB ($2^{32}$ bytes) limit on various things (uncompressed size of a file, compressed size of a file, and total size of the archive), as well as a limit of 65,535 ($2^{16}$-1) entries in a ZIP archive. In version 4.5 of the specification (which is not the same as v4.5 of any particular tool), PKWARE introduced the "ZIP64" format extensions to get around these limitations, increasing the limits to 16 EB ($2^{64}$ bytes). In essence, it uses a "normal" central directory entry for a file, followed by an optional "zip64" directory entry, which has the larger fields.[37]

The format of the Local file header (LOC) and Central directory entry (CEN) are the same in ZIP and ZIP64. However, ZIP64 specifies an extra field that may be added to those records at the discretion of the compressor, whose purpose is to store values that do not fit in the classic LOC or CEN records. To signal that the actual values are stored in ZIP64 extra fields, they are set to 0xFFFF or 0xFFFFFFFF in the corresponding LOC or CEN record.

**Zip64 extended information extra field**

| Offset | Bytes | Description[31] |
|---|---|---|
| 0 | 2 | Header ID 0x0001 |
| 2 | 2 | Size of the extra field chunk (8, 16, 24 or 28) |
| 4 | 8 | Original uncompressed file size |
| 12 | 8 | Size of compressed data |
| 20 | 8 | Offset of local header record |
| 28 | 4 | Number of the disk on which this file starts |

On the other hand, the format of EOCD for ZIP64 is slightly different from the normal ZIP version.[31]

**Zip64 End of central directory record (EOCD64)**

| Offset | Bytes | Description[31] |
|---|---|---|
| 0 | 4 | End of central directory signature = 0x06064b50 |
| 4 | 8 | Size of the EOCD64 minus 12 |
| 12 | 2 | Version made by |

| | | |
|---|---|---|
| 14 | 2 | Version needed to extract (minimum) |
| 16 | 4 | Number of this disk |
| 20 | 4 | Disk where central directory starts |
| 24 | 8 | Number of central directory records on this disk |
| 32 | 8 | Total number of central directory records |
| 40 | 8 | Size of central directory (bytes) |
| 48 | 8 | Offset of start of central directory, relative to start of archive |
| 56 | *n* | Comment (up to the size of EOCD64) |

It is also not necessarily the last record in the file. A End of Central Directory Locator follows (an additional 20 bytes at the end).

The File Explorer in Windows XP does not support ZIP64, but the Explorer in Windows Vista and later do.[*citation needed*] Likewise, some extension libraries support ZIP64, such as DotNetZip, QuaZIP[38] and IO::Compress::Zip in Perl. Python's built-in zipfile supports it since 2.5 and defaults to it since 3.4.[39] OpenJDK's built-in java.util.zip supports ZIP64 from version Java 7.[40] Android Java API support ZIP64 since Android 6.0.[41] Mac OS Sierra's Archive Utility notably does not support ZIP64, and can create corrupt archives when ZIP64 would be required.[42] However, the ditto command shipped with Mac OS will unzip ZIP64 files.[43] More recent[*when?*] versions of Mac OS ship with info-zip's zip and unzip command line tools which do support Zip64: to verify run zip -v and look for "ZIP64_SUPPORT".

## Combination with other file formats[edit]

The `.ZIP` file format allows for a comment containing up to 65,535 ($2^{16}-1$) bytes of data to occur at the end of the file after the central directory.[31] Also, because the central directory specifies the offset of each file in the archive with respect to the start, it is possible for the first file entry to start at an offset other than zero, although some tools, for example gzip, will not process archive files that do not start with a file entry at offset zero.

This allows arbitrary data to occur in the file both before and after the ZIP archive data, and for the archive to still be read by a ZIP application. A side-effect of this is that it is possible to author a file that is both a working ZIP archive and another

format, provided that the other format tolerates arbitrary data at its end, beginning, or middle. Self-extracting archives (SFX), of the form supported by WinZip, take advantage of this, in that they are executable (`.exe`) files that conform to the PKZIP AppNote.txt specification, and can be read by compliant zip tools or libraries.

This property of the `.ZIP` format, and of the JAR format which is a variant of ZIP, can be exploited to hide rogue content (such as harmful Java classes) inside a seemingly harmless file, such as a GIF image uploaded to the web. This so-called GIFAR exploit has been demonstrated as an effective attack against web applications such as Facebook.[44]

## Limits[edit]

The minimum size of a `.ZIP` file is 22 bytes. Such an *empty zip file* contains only an End of Central Directory Record (EOCD):

```
[0x50,0x4B,0x05,0x06,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00]
```

The maximum size for both the archive file and the individual files inside it is 4,294,967,295 bytes ($2^{32}-1$ bytes, or 4 GB minus 1 byte) for standard ZIP. For ZIP64, the maximum size is 18,446,744,073,709,551,615 bytes ($2^{64}-1$ bytes, or 16 EB minus 1 byte).[45]

## Proprietary extensions[edit]

### Extra field[edit]

`.ZIP` file format includes an extra field facility within file headers, which can be used to store extra data not defined by existing ZIP specifications, and which allow compliant archivers that do not recognize the fields to safely skip them. Header IDs 0–31 are reserved for use by PKWARE. The remaining IDs can be used by thirdparty vendors for proprietary usage.

#### Strong encryption controversy[edit]

When WinZip 9.0 public beta was released in 2003, WinZip introduced its own AES256 encryption, using a different file format, along with the documentation for the new specification.[46] The encryption standards themselves were not proprietary, but PKWARE had not updated APPNOTE.TXT to include Strong Encryption Specification (SES) since 2001, which had been used by PKZIP versions 5.0 and 6.0. WinZip technical consultant Kevin Kearney and StuffIt product manager Mathew Covington accused PKWARE of withholding SES, but PKZIP chief technology officer Jim Peterson claimed that certificate-based encryption was still incomplete.

In another controversial move, PKWare applied for a patent on 16 July 2003 describing a method for combining ZIP and strong encryption to create a secure file.[47]

In the end, PKWARE and WinZip agreed to support each other's products. On 21 January 2004, PKWARE announced the support of WinZip-based AES compression format.[48] In a later version of WinZip beta, it was able to support SES-based ZIP files.[49] PKWARE eventually released version 5.2 of the .ZIP File Format Specification to the public, which documented SES. The Free Software project 7-Zip also supports AES, but not SES in ZIP files (as does its POSIX port p7zip).

When using AES encryption under WinZip, the compression method is always set to 99, with the actual compression method stored in an AES extra data field.[50] In contrast, Strong Encryption Specification stores the compression method in the basic file header segment of Local Header and Central Directory, unless Central Directory Encryption is used to mask/encrypt metadata.

## Implementation[edit]

There are numerous .ZIP tools available, and numerous .ZIP libraries for various programming environments; licenses used include proprietary and free software. WinZip, WinRAR, Info-ZIP, 7-Zip, PeaZip and B1 Free Archiver are wellknown .ZIP tools, available on various platforms. Some of those tools have library or programmatic interfaces.

Some development libraries licensed under open source agreement are libzip, libarchive, and Info-ZIP. For Java: Java Platform, Standard Edition contains the package "java.util.zip" to handle standard .ZIP files; the Zip64File library specifically supports large files (larger than 4 GB) and treats .ZIP files using random access; and the Apache Ant tool contains a more complete implementation released under the Apache Software License.

The Info-ZIP implementations of the .ZIP format adds support for Unix filesystem features, such as user and group IDs, file permissions, and support for symbolic links. The Apache Ant implementation is aware of these to the extent that it can create files with predefined Unix permissions. The Info-ZIP implementations also know how to use the error correction capabilities built into the .ZIP compression format. Some programs do not, and will fail on a file that has errors.

The Info-ZIP Windows tools also support NTFS filesystem permissions, and will make an attempt to translate from NTFS permissions to Unix permissions or vice versa when extracting files. This can result in potentially unintended combinations, e.g. .exe files being created on NTFS volumes with executable permission denied.

Versions of Microsoft Windows have included support for .ZIP compression in Explorer since the Microsoft Plus! pack was released for Windows 98. Microsoft calls this feature "Compressed Folders". Not all .ZIP features are supported by the Windows Compressed Folders capability. For example, encryption is not supported in Windows 10 Home edition,[51] although it can decrypt. Unicode entry encoding is not supported until Windows 7, while split and spanned archives are not readable or writable by the Compressed Folders feature, nor is AES Encryption supported.[52]

Microsoft Office started using the zip archive format in 2006 for their Office Open XML .docx, .xlsx, .pptx, etc. files, which became the default file format with Microsoft Office 2007.

## Legacy[edit]

There are numerous other standards and formats using "zip" as part of their name. For example, zip is distinct from gzip, and the latter is defined in IETF RFC 1952. Both zip and gzip primarily use the DEFLATE algorithm for compression. Likewise, the ZLIB format (IETF RFC 1950) also uses the DEFLATE compression algorithm, but specifies different headers for error and consistency checking. Other common, similarly named formats and programs with different native formats include 7Zip, bzip2, and rzip.

## Concerns[edit]

The theoretical maximum compression factor for a raw DEFLATE stream is about 1032 to one,[53] but by exploiting the ZIP format in unintended ways, ZIP archives with compression ratios of billions to one can be constructed. These zip bombs unzip to extremely large sizes, overwhelming the capacity of the computer they are decompressed on.[54]

## See also[edit]

- Comparison of file archivers
- Comparison of archive formats
- List of archive formats

## References[edit]


1. ^ Jump up to:[a] [b] [c] *Registration of a new MIME Content-Type/Subtype - application/zip, IANA, 20 July 1993, retrieved 5 January 2012*
2. ^ *"Phillip Katz, Computer Software Pioneer, 37". The New York Times'. 1 May 2000. Retrieved 14 June 2009.*
3. ^ *Murray, Matt; Tannenbaum, Jeffrey A. (15 August 1997). "The Rise and Fall of a Software Star; Phil Katz Loved Code -- and Liquor". The Wall Street Journal (online ed.). Archived from the original on 4 March 2016.* Alt URL Updated 2000-06-19.
4. ^ Jump up to:[a] [b] *"The BBS Documentary Library". www.bbsdocumentary.com. Retrieved 25 September 2020.*
5. ^ Jump up to:[a] [b] *Stay, Michael. "ZIP Attacks with Reduced Known Plaintext" (PDF). Math.ucr.edu. Archived from the original (PDF) on 28 October 2017. Retrieved 9 September 2017.*
6. ^ *Brian Livingston (8 September 2003), PKZip Must Open Up, retrieved 5 January 2012, The ZIP file format is given freely into the public domain and can be claimed neither legally nor morally by any individual, entity or company*
7. ^ *Where Did Zip Files Come From Anyway?, Infinity Design Concepts, Inc., retrieved 5 January 2012*
8. ^ *Press Release, 1989, retrieved 5 January 2012*
9. ^ *Our Founder - Phil Katz, PKWARE, archived from the original on 1 October 2010, retrieved 5 January 2012*
10. ^ *Gareth Horton; Rob Weir; Alex Brown (2 November 2010), sc34-wg1, retrieved 5 January 2012*
11. ^ *.ZIP Application Note, retrieved 20 July 2012*

12. ^ *File: APPNOTE.TXT - .ZIP File Format Specification Version: 4.5 Revised: 11/01/2001*, 3 December 2001, archived from the original on 3 December 2001, retrieved 21 April 2012
13. ^ *APPNOTE.TXT - .ZIP File Format Specification, Version: 5.2 - Notification of Change*, 16 July 2003, retrieved 5 January 2012
14. ^ *File: APPNOTE.TXT - .ZIP File Format Specification Version: 5.2 - Notification of Change – Revised: 06/02/2003*, 2 July 2003, archived from the original on 2 July 2003, retrieved 21 April 2012
15. ^ *File: APPNOTE - .ZIP File Format Specification Version: 6.1.0 - Notification of Change – Revised: 01/20/2004*, 19 August 2004, archived from the original on 19 August 2004, retrieved 21 April 2012
16. ^ *APPNOTE.TXT - .ZIP File Format Specification, Version: 6.2.0 - Notification of Change*, 26 April 2004, retrieved 5 January 2012
17. ^ *APPNOTE.TXT - .ZIP File Format Specification, Version: 6.3.0*, 29 September 2006, retrieved 5 January 2012
18. ^ *APPNOTE.TXT - .ZIP File Format Specification, Version: 6.3.1*, 11 April 2007, retrieved 25 June 2018
19. ^ *APPNOTE.TXT - .ZIP File Format Specification Version: 6.3.2*, 28 September 2007, retrieved 25 June 2018
20. ^ *APPNOTE.TXT - .ZIP File Format Specification Version: 6.3.3*, 1 September 2012, retrieved 25 June 2018
21. ^ *APPNOTE.TXT - .ZIP File Format Specification Version: 6.3.4*, 1 October 2014, retrieved 25 June 2018
22. ^ Jump up to: [a] [b] [c] *APPNOTE.TXT - .ZIP File Format Specification Version: 6.3.5*, 20 December 2018, retrieved 3 January 2019
23. ^ *APPNOTE.TXT - .ZIP File Format Specification Version: 6.3.6*, 26 April 2019, retrieved 3 January 2019
24. ^ *APPNOTE.TXT - .ZIP File Format Specification Version: 6.3.7*, 1 June 2020, retrieved 6 June 2020
25. ^ *APPNOTE.TXT - .ZIP File Format Specification Version: 6.3.8*, 15 June 2020, retrieved 7 July 2020
26. ^ *APPNOTE.TXT - .ZIP File Format Specification Version: 6.3.9*, 15 July 2020, retrieved 8 August 2020
27. ^ *"Additional Compression Methods Specification"*. WinZip. Mansfield, CT: WinZip Computing, S.L. 19 May 2009. Retrieved 24 May 2009.
28. ^ *"What is a Zipx File?"*. Winzip: Knowledgebase. Mansfield, CT: WinZip Computing, S.L. 13 August 2010. Retrieved 17 August 2010.
29. ^ *"ISO/IEC JTC 1/SC 34 — Document Description and Processing Languages"* (PDF). 12 April 2010. Archived from the original (PDF) on 12 May 2014. Retrieved 10 May 2014.
30. ^ *"ISO/IEC 21320-1:2015 Document Container File — Part 1: Core"*. ITTF. 2015.
31. ^ Jump up to: [a] [b] [c] [d] [e] [f] [g] [h] [i] *"File : APPNOTE.TXT - .ZIP File Format Specification : Version: 6.3.4"* (TXT). Pkware.com. Retrieved 9 September 2017.
32. ^ *"File: APPNOTE.TXT - .ZIP File Format Specification"*. PKWARE Inc. Retrieved 21 February 2022.
33. ^ Adler, Mark. *"How are zlib, gzip and zip related? What do they have in common and how are they different?"*. Retrieved 27 November 2018.
34. ^ *"Frequently Asked Questions about zlib"*. zlib. The PKWare DCL uses a completely different compressed data format than does PKZIP and zlib. However, you can look in zlib's contrib/blast directory for a possible solution to your problem. (contrib/blast)
35. ^ *"AES Encryption Information: Encryption Specification AE-1 and AE-2"*. Winzip.com. Retrieved 9 September 2017.
36. ^ *"APPNOTE - PKZIP/SecureZIP - PKWARE Support Site"*. Pkware.com. Retrieved 9 September 2017.
37. ^ *"File : APPNOTE.TXT - .ZIP File Format Specification : Version: 6.3.4"* (TXT). Pkware.cachefly.net. Retrieved 9 September 2017.
38. ^ *"QuaZIP changes"*. 22 January 2014. Retrieved 25 January 2014.
39. ^ *"Python enhancement: Use allowZip64=True by default (3.4)"*. Retrieved 6 May 2014.
40. ^ Shen, Xueming (17 April 2009). *"ZIP64, The Format for > 4G Zipfile, Is Now Supported"*. Xueming Shen's Blog. Sun Microsystems. Retrieved 27 September 2010.
41. ^ *"Sign in - Google Accounts"*. code.google.com. Retrieved 9 September 2017.

42. ^ *"Error: invalid central directory file header signature when unzipping big files, zipped by mac os · Issue #69 · thejoshwolfe/yauzl"*. GitHub.
43. ^ *"Extract large zip file (50 GB) on Mac OS X"*. Retrieved 17 December 2018.
44. ^ McMillan, Robert (August 2008). *"A photo that can steal your online credentials"*. Infoworld.com. Retrieved 9 September 2017.
45. ^ *"ZipArchive: Zip64 Format: Crossing the Limits of File Sizes and Number of Files and Segments"*. Artpol-software.com. Retrieved 9 September 2017.
46. ^ *"WinZip – AES Encryption Information"*. Winzip.com. Retrieved 9 September 2017.
47. ^ McMillan, Robert (25 July 2003). *"PKWare seeks patent for .zip file format"*. InfoWorld.com. Archived from the original on 10 August 2003. Retrieved 16 June 2008.
48. ^ *"Software makers patch Zip tiff"*. News.com. Retrieved 9 September 2017.
49. ^ John Leyden. *"Zip file encryption compromise thrashed out"*. Theregister.co.uk. Retrieved 9 September 2017.
50. ^ *"AES Encryption Information: Encryption Specification AE-1 and AE-2"*. Winzip.com. Retrieved 9 September 2017.
51. ^ Maham Mukhtar (August 2017). *"2 Ways To Fix "Encrypt Contents To Secure Data" Option Grayed Out In Windows 10"*. iTechtics. EFS is available for all editions of Windows 10 except Windows 10 Home edition.
52. ^ *"Why is Windows Compressed Folders (Zip folders) support stuck at the turn of the century?"*. 15 May 2018.
53. ^ *"zlib Technical Details"*. Retrieved 10 July 2019.
54. ^ Smith, Ernie (10 July 2019). *"The Most Clever 'Zip Bomb' Ever Made Explodes a 46MB File to 4.5 Petabytes"*. Motherboard. Vice Media. Retrieved 10 July 2019.

## External links[edit]

- .ZIP Application Note landing page for PKWARE's current and historical .ZIP file
- ISO/IEC 21320-1:2015 — Document Container File — Part 1: Core
- Zip Files: History, Explanation and Implementation
- Shrink, Reduce, and Implode: The Legacy Zip Compression Methods
- APPNOTE.TXT mirror
- Structure of PKZip file Format specifications, graphical tables